NFL Matchup Predictor

EECS 349, Northwestern University Spring 2016

Thornton Uhl (thorntonuhl2018@u.northwestern.edu)
Daniel Thomas (danielthomas2018@u.northwestern.edu)
Scott Renshaw (scottrenshaw2018@u.northwestern.edu)

Introduction

American football is one of the hardest sports to predict in the world. There is an astounding range of factors which determine the outcome of each game - including player statistics, weather, stadium environment, etc. Being able accurately predict NFL games has a huge range of implications with regard to betting, sports management, sports analytics, etc. Thus, our question for this project is as follows: *Given any matchup of two teams on a given week, can we predict the winner correctly?* Because of the nature of NFL games, the best predictors do not perform with better than 72% accuracy. As such, our goal was to get as close to this figure as possible - hopefully within 5% of it.

Process

The first (and by far the most time-intensive) part of this project was data compilation. Because of the nature of the required data, there were no complete data sets available. However, we were able to collect smaller data sets from pro-football-reference.com. These data sets listed team statistics for all 32 NFL franchises in game-by-game format. We downloaded these CSV data sets, which had the following attributes for each game: Location, Opponent, Points Scored, Points Allowed, First Downs, First Downs Allowed, Total Yards Gained, Total Yards Allowed, Passing Yards Gained, Passing Yards Allowed, Rushing Yards Gained, Rushing Yards Allowed, Average Turnovers, Average Turnovers Allowed, Expected Offensive Points, Expected Defensive Points, and Expected Special Teams Points. We imported this data into Excel and used Excel functions to convert the data into moving averages. For week 1, we imported averages from the previous season. For example, in predicting week 3 matchups, we used an average of the team's statistics from the previous season, week 1, and week 2. We then had 32 distinct data sets - one for each NFL franchise. We compiled the data for all 32 teams into a single CSV file, with (32 teams)*(16 games per team) = 512 instances. Afterward, this entire process was repeated such that we had a training set (every matchup from the 2014-2015 NFL season) and a test set (every matchup from the 2015-2016 NFL season).

Our initial data sets had no attributes besides pure NFL statistics. We ran 12 classifiers on our data, which produced these 10-fold accuracies in Weka:

ZeroR	J48	IBk	Multilayer Perceptron	Decision Table	Naive Bayes
49.2188%	57.4219%	52.1484%	57.4219%	62.1094%	55.2734%

Logistic	Bayes Net	JRip	KStar	Conjunctive Rule	SMO
61.1328%	62.5%	53.3203%	50.3906%	56.6406%	64.0625%

SMO was the best performing classifier on our initial Weka tests. (Also note that Bayes Net and Decision Table were among the best performing classifiers - an early indication of the usefulness for DTNB for this project, although Naive Bayes was an average performer.)

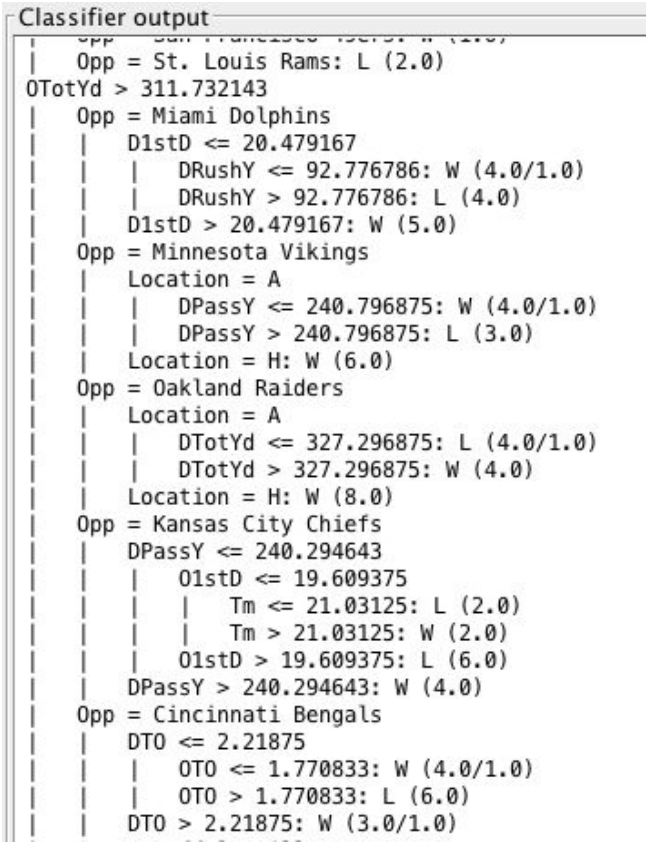
However, because football is so unpredictable of a sport, we sought to collect more statistics for inclusion in our data set. We felt that we might be able to increase our performance if we could numerically capture a team's confidence - it's week-by-week momentum. Confidence/momentum is a huge factor in determining the outcome of

a given matchup, but it is very difficult to capture numerically. We sought to create an expression for momentum that could get our accuracy closer to 72%.

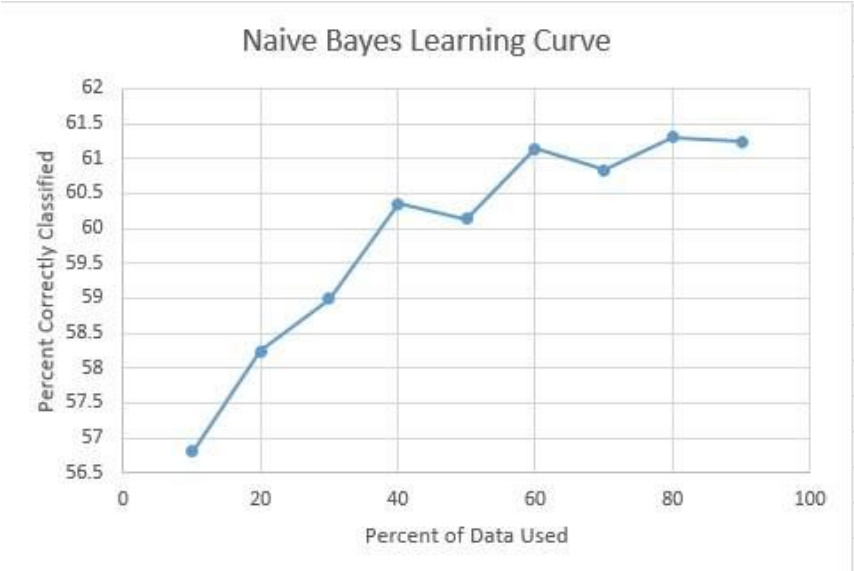
In order to create this expression, we first had to determine which attributes were most important. We did this by examining early branches of pruned J48 trees on Weka. Predictably, point scored and points allowed were very commonly atop the list. However - more interestingly - we noticed that turnovers also extremely telling. Teams that turned the ball a few times a game, on average, were performing considerably worse than other teams. As such, we developed an expression for momentum by including these two important characteristics. We ran tons of tests in Weka with slightly different data, each time making a small adjustment to the expression. Eventually we got the best results using this expression:

Momentum(week n) = [Tm(n-3)+Tm(n-2)+Tm(n-1)-Opp(n-3)-Opp(n-2)-Opp(n-1)]-[DTO(n-3)+DTO(n-2)+DTO(n-1)-OTO(n-3)-OTO(n-2)-OTO(n-1)]*100

This momentum function improved accuracy among the majority of the tests, including a 1.9531% Logistic accuracy increase. Because we were only trying to get closer to 72%, we were pleased with a ~2% accuracy increase. As such, we turned to Python and wrote a program with functions that would introduce new momentum/confidence attributes and deal with all the Excel/CSV manipulation itself. In the meantime, we continued to test on Weka and noticed that DTNB was consistently outperforming all other classifiers (65.2344% on our new data that included the Momentum attribute.) We did some research about the Decision Table-Naive Bayes hybrid and understood that because of the nature of our data, DTNB would continue to consistently yield the highest accuracy numbers. For any given team and week, one of the most important attributes to consider is a team's opponent. However, the large majority of our data is numerical/statistical. Because the optimal classification occurs by examining one extremely important categorical attribute and numerous numerical attributes, DTNB is very well-suited for our project (and any similar sports matchup predictor, for that matter.) This is easily observable by examining J48 on Weka:



Observe that for J48, Opponent is one of the first splits, and then *different* numerical attributes are considered, given each opponent. This supported our decision to use strictly DTNB for the remainder of the project. Until this point, we had been constantly playing with Weka and the scikit-learn module on Python with SciPy, NumPy, and matplotlib. However, because of the nature of our tree and the Naive Bayes learning curve (see below) it was clear that our data was well-suited for DTNB, and we would not achieve higher accuracy with other classification methods. Using the object editor in Weka, we found that the best function was "DTNB -X 1".



Once our Python code was finished, we were able to supplement the data with extra analytical attributes. The idea was to input the data of statistical averages for each team's season, game-by-game, and use the code to highlight patterns. One potential information source we checked was the derivative of some statistics with respect to each game as a time step. Also, we built off the momentum expression that gave us the ~2% Logistic accuracy increase, attempting to enumerate the potential for a team to continue a "hot streak" in a particular statistical category from game to game. This often involved differentials of a particular statistic with between a given team and the average of its recent opponents. We added ten attributes with analyze.py. The datasets, the code, and comments within the python file describing each attribute can be found on our github repository[1]. The immediate addition of these ten attributes actually lowered our DTNB accuracy. We then tested data sets with just one attribute, iterating through which attribute it would be, and increasing the number of attributes we used. We observed the optimal performance when we excluded yard_differential and included scoring_momentum3, scoring_momentum4, w_l. We found that the other attributes were not used and therefore didn't impact the accuracy.

Conclusion[2]

The optimal attribute selection using analyze.py yielded an accuracy of 66.9922%. This was just .0078 below our initial goal of getting within 5% of the current best NFL predictors. We benefited from DTNB as a classifier (initial accuracy ~65%) and then added another ~2% accuracy by defining and coding our own mathematical expressions. One interesting thing we learned from our testing was that yard differentials have essentially no effect on game outcome. Turnovers, for example, were far more important. In other words, it doesn't matter how far a team drives downfield if they fail to convert in the Red Zone or turn the ball over.

---

[1] https://github.com/danielrthomas/EECS349_NFL_Predictor

[2] Work distribution: Typically all three of us were present at our group meetings, so most work was done together. That said, Thornton and Scott generally focused on the data compilation, Weka testing, and scikit-learn, and Dan focused on attribute analysis through Python.